Amendments to the claims,
Listing of all claims pursuant to 37 CFR 1.121(c)

This listing of claims will replace all prior versions, and listings, of claims in the application:

What is claimed is:

1. (Currently amended) In a database system employing a transaction log, an improved method for restoring databases to a consistent version supporting read-only uses, the method comprising:

providing a shared cache storing database blocks in memory of the database system;

creating a write view of a given database in the shared cache supporting read and write uses of the given database;

in response to a read-only transaction of ~~a~~the given database, creating a read-only cache view of the given database using the given database's transaction log by logically undoing transactions which have begun but have yet to commit, said read-only cache view comprising particular database blocks in the shared cache that record a transactionally consistent version of the given database at a given point in time supporting read-only uses;

temporarily storing any database blocks that overflow said shared cache in a temporary database during use of the read-only cache view by the read-only transaction; and

performing the read-only transaction using the read-only cache view and returning results of the read-only transaction, without blocking performance of transactions involving write operations using the write view of the given database.

2. (Currently amended) The method of claim 1, wherein during occurrence of the read-only transaction any database blocks associated with the read-only cache view are not written from the shared cache to the given database.

3. (Previously presented) The method of claim 1, wherein the blocks that

overflow the shared cache are temporarily stored in a temporary database.

4. (Canceled)

5. (Currently amended) The method of claim 1, wherein the temporary database is used only in the event the read-only cache view overflows the shared cache.

6. (Previously presented) The method of claim 1, further comprising:

using an allocation bitmap for indicating database blocks temporarily stored in the temporary database.

7. (Previously presented) The method of claim 6, further comprising:

upon completion of the read-only transaction, deleting the blocks temporarily stored in the temporary database by updating the allocation bitmap.

8. (Currently amended) The method of claim 1, wherein said step of temporarily storing database blocks overflowing the shared cache includes storing a mapping to said database blocks in a table in the temporary database including a first column for maintaining a block number of a read-only cache view block having undo/redo records applied to it and a second column for maintaining a block number allocated to temporarily store said database blocks overflowing the shared cache.

9. (Canceled)

10. (Currently amended) The method of claim 1, further comprising:

upon termination of the read-only transaction, marking the read-only cache view as closed.

11. (Currently amended) The method of claim 10, further comprising:

when new block allocations need to be made in the shared cache, traversing the shared cache looking for database blocks to purge; and

purging database blocks from any read-only cache view that have been marked as closed.

12. (Currently amended) The method of claim 1, further comprising:

sharing the read-only cache view created for the read-only transaction with other read-only transactions which start within a specified period of time following the start of the read-only transaction.

13. (Previously presented) The method of claim 1, further comprising:

detecting the read-only transaction; and

upon occurrence of write operations, adding back link log records to the database's transaction log that serve to link together log records of the transaction log that pertain to a write transaction.

14. (Previously presented) The method of claim 13, further comprising:

if the read-only transaction must be undone, using the back link log records to skip portions of the transaction log that are irrelevant for undoing an uncommitted write transaction, wherein the back link log records are only generated in the transaction log when there are active read only transactions.

15. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 1.

16. (Previously presented) The method of claim 1, further comprising:

downloading a set of processor-executable instructions for performing the method of claim 1.

17. (Currently amended) A database system capable of restoring databases to a consistent version supporting read-only uses, the system comprising:

a computer having a processor and memory;

a log manager module which manages a transaction log of the database system;

a cache manager module for managing a shared cache that stores database blocks in memory of the database system and creating write view of a given database in the shared cache supporting read and write uses of the given database and a read-only cache view of a given database using the transaction log of the given database, said read-only cache view being created in response to a read-only transaction of the given database, said read-only cache view comprising particular database blocks of the shared cache that record a view of a particular version of the given database at a given point in time, wherein the cache manager utilizes a temporary database for storing any database blocks that overflow said shared cache during use of the read-only cache view by the read-only transaction and invokes a transaction manager module for logically undoing transactions which have begun but have yet to commit in creating the read-only cache view; and

a transaction manager module for logically undoing transactions which have begun but have yet to commit upon starting the read-only transaction in order to construct the read-only cache view comprising a transactionally consistent prior version of the given database, performing the read-only transaction using the read-only cache view without blocking performance of transactions involving write operations using the write view of the given database, and returning results of the read-only transaction.

18. (Currently amended) The system of claim 17, wherein during occurrence of the read-only transaction any database blocks associated with the read-only cache view are not written from the shared cache to the given database.

19. (Previously presented) The system of claim 17, wherein the cache manager stores database blocks that overflow the shared cache in a temporary database.

20. (Canceled)

21. (Currently amended) The system of claim 17, wherein the temporary database is used only in the event the read-only cache view overflows the shared cache.

22. (Previously presented) The system of claim 17, wherein said cache manager

maintains an allocation bitmap indicating database blocks temporarily stored in the temporary database.

23. (Previously presented) The system of claim 22, wherein said cache manager deletes blocks from the temporary database by updating the allocation bitmap.

24. (Currently amended) The system of claim 17, wherein said cache manager stores a mapping to database blocks overflowing the shared cache in a table of the temporary database including a first column for maintaining a block number of a read-only cache view block having undo/redo records applied to it and a second column for maintaining a block number allocated to temporarily store said database blocks overflowing the shared cache.

25. (Canceled)

26. (Currently amended) The system of claim 17, wherein said cache manager marks the read-only cache view as closed, upon termination of the read-only transaction.

27. (Currently amended) The system of claim 26, wherein said cache manager traverses the shared cache looking for database blocks to purge, and purges database blocks from any read-only cache view that have been marked as closed when new block allocations need to be made in the shared cache.

28. (Currently amended) The system of claim 17, wherein said cache manager shares the read-only cache view created for the read-only transaction with other read-only transactions which start within a specified period of time following the start of the read-only transaction.

29. (Previously presented) The system of claim 17, wherein said log manager detects the read-only transaction, and adds back link log records to the transaction log that serve to link together log records of the transaction log that pertain to a write

transaction that may need to be logically undone.

30. (Previously presented) The system of claim 29, wherein said log manager uses the back link log records to skip portions of the transaction log that are irrelevant for undoing the write transaction, wherein the back link log records are only generated in the transaction log when there are active read only transactions.